Task:

Find the symmetric words for the given review text

In [1]:

```
4/mwA7fZ3oCpQf7dp46SepqAxJOo686mP1PRZoChEs1T5YTiFB6Ar9eUI# Load the Drive helper and mount
from google.colab import drive

# This will prompt for authorization.
drive.mount('/content/drive')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6
qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.com&redirect_uri=urn%3Aietf%3Awg%3Aoauth%3A2.0%
b&scope=email%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdocs.test%20https%3A%2F%2Fwww.googleapis.
2Fauth%2Fdrive%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdrive.photos.readonly%20https%3A%2F%2Fww
ogleapis.com%2Fauth%2Fpeopleapi.readonly&response_type=code

Enter your authorization code:
··········
Mounted at /content/drive
```

In [22]:

```
%env JOBLIB_TEMP_FOLDER=/tmp
```

```
env: JOBLIB_TEMP_FOLDER=/tmp
```

In [23]:

```
import warnings
warnings.filterwarnings("ignore")
import numpy as np
import pandas as pd

final = pd.read_csv('drive/My Drive/Colab Notebooks/matrix-factor/review.csv')
final.head(4)
```

Out[23]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator	Score	Time
0	1	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	1	1	5	130386240(
1	2	B00813GRG4	A1D87F6ZCVE5NK	dll pa	0	0	1	134697600(
2	3	B000LQOCH0	ABXLMWJIXXAIN	Natalia Corres "Natalia Corres"	1	1	4	121901760(

3	4	B000UA0QIQ	A395BORC6FGVXV	Karl	3	3	2	1307923200
	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator	Score	Time

After cleaning the text

In [24]:

```
import pickle
final = pickle.load(open('drive/My Drive/Colab Notebooks/cluster/final.p','rb'))
final.head(4)
```

Out[24]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator	Score	
138706	150524	0006641040	ACITT7DI6IDDL	shari zychinski	0	0	1	93
138683	150501	0006641040	AJ46FKXOVC7NR	Nicholas A Mesiano	2	2	1	94
417839	451856	B00004CXX9	AIUWLEQ1ADEG5	Elizabeth Medina	0	0	1	94
346055	374359	B00004CI84	A344SMIA5JECGM	Vincent P. Ross	1	2	1	94

Fetching top 2000 IDF words from TFIDF

In [0]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import CountVectorizer
from pandas import DataFrame
import pandas as pd

X=final['CleanedText'].iloc[0:100000]

m = TfidfVectorizer(max_features=2000)
tf_idf_matrix = m.fit_transform(X)
# we are converting a dictionary with word as a key, and the idf as a value
dictionary = dict(zip(m.get_feature_names(), list(m.idf_)))

a= list(dictionary.keys())
```

Defining Co-occurrence matrix

In [0]:

```
def co_mat(ctxs,l_unique):
  l_unique = list(set((' '.join(ctxs)).split(' ')))


  mat = np.zeros((len(l_unique), len(l_unique)))
```

```
  mat = np.zeros((len(l_unique), len(l_unique)))

  nei = []
  nei_size = 5

  for ctx in ctxs:
        words = ctx.split(' ')

        for i, _ in enumerate(words):
          nei.append(words[i])

          if len(nei) > (nei_size * 2) + 1:
              nei.pop(0)

          pos = int(len(nei) / 2)
          for j, _ in enumerate(nei):
              mat[l_unique.index(nei[j]), l_unique.index(words[i])] += 1

  mat = pd.DataFrame(mat)
  mat.index = l_unique
  mat.columns = l_unique

  return mat
```

In [31]:

```
import numpy as np

mat= co_mat(a,X)
mat.shape
```

Out[31]:

```
(2000, 2000)
```

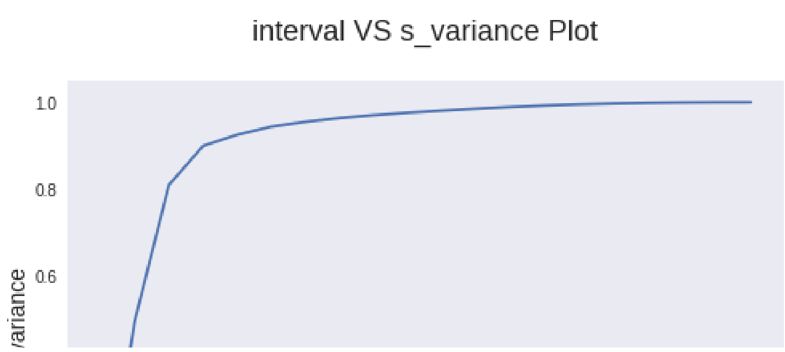
Truncated SVD part

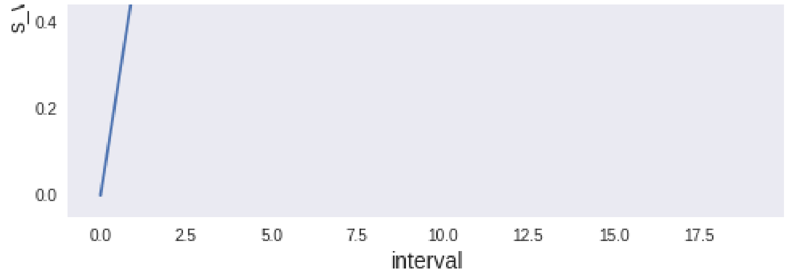
In [0]:

```
from sklearn.decomposition import TruncatedSVD

s_variance =[]
interval= []
for i,end in enumerate(range(0,1999,100)):
    tsvd= TruncatedSVD(n_components=end).fit(mat)
    interval.append(i)
    s_variance.append(tsvd.explained_variance_ratio_.sum())
```

In [0]:

```
# Loss plot
import matplotlib.pyplot as plt
# Draw Loss VS K values plot
plt.plot(interval,s_variance)
plt.xlabel('interval',size=14)
plt.ylabel('s_variance',size=14)
plt.title('interval VS s_variance Plot\n',size=18)
plt.grid()
plt.show()
```





Choosing 5th interval (5*100)

In [34]:

```
# Choosing 500 as max informaton dimenson
from sklearn.decomposition import TruncatedSVD

tsvd1= TruncatedSVD(n_components=500)
data=tsvd1.fit_transform(mat)

tsvd1.explained_variance_ratio_.sum()
```

Out[34]:

```
0.9443946602091106
```

Gives 94% of data at 500th dimension

In [35]:

```
tsvd1.components_.shape
```

Out[35]:

```
(500, 2000)
```

Kmeans++ clusters of 50 words

In [9]:

```
from sklearn.cluster import KMeans

clf = KMeans(n_clusters = 50, n_init = 8, n_jobs = -1)
clf.fit(data)
```

Out[9]:

```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
    n_clusters=50, n_init=8, n_jobs=-1, precompute_distances='auto',
    random_state=None, tol=0.0001, verbose=0)
```

In [10]:

```
# Randomly choosed clusters
cluster0 = []
cluster30 = []
cluster49 = []
check =[]

for i in range(clf.labels_.shape[0]):
    if clf.labels_[i] == 0:
        cluster0.append(a[i])

    elif clf.labels_[i] == 30:
        cluster30.append(a[i])

    elif clf.labels_[i] == 49:
        cluster49.append(a[i])

    else:
```

```
    else:
        check.append(a[i])



print("\nNo. of reviews in Cluster-0 : ",len(cluster0))
print("\nNo. of reviews in Cluster-30 : ",len(cluster30))
print("\nNo. of reviews in Cluster-49 : ",len(cluster49))
print("\nNo. of reviews in rest : ",len(check))
```

```
No. of reviews in Cluster-0 :  33

No. of reviews in Cluster-30 :  29

No. of reviews in Cluster-49 :  24

No. of reviews in rest :  1914
```

Randomly choosed clusters

In [0]:

```
cluster0
```

Out[0]:

```
['acai',
 'allergi',
 'carb',
 'cereal',
 'chai',
 'comparison',
 'complain',
 'creami',
 'cupboard',
 'detail',
 'duck',
 'exclus',
 'favorit',
 'film',
 'fire',
 'grade',
 'gum',
 'haribo',
 'kernel',
 'locat',
 'moment',
 'move',
 'non',
 'onlin',
 'past',
 'pet',
 'pineappl',
 'purs',
 'seven',
 'splash',
 'subscrib',
 'theyd',
 'werent',
 'what',
 'yellow']
```

In [0]:

```
cluster30
```

Out[0]:

```
['bargain',
 'bergamot',
 'bread',
 'break',
 'breast',
 'car',
```

```
 'cheaper',
 'chef',
 'clove',
 'cracker',
 'econom',
 'even',
 'gift',
 'glycem',
 'gotta',
 'gravi',
 'later',
 'lighter',
 'mountain',
 'mushi',
 'pair',
 'penni',
 'plus',
 'secret',
 'shock',
 'soda',
 'sour',
 'steak',
 'toffe',
 'tofu',
 'tortilla',
 'trick',
 'zero']
```

In [0]:

```
cluster49
```

Out[0]:

```
[]
```

Word cloud

In [11]:

```
!pip install WordCloud
```

```
Collecting WordCloud
  Downloading
https://files.pythonhosted.org/packages/ae/af/849edf14d573eba9c8082db898ff0d090428d9485371cc4fe21a6
ad2/wordcloud-1.5.0-cp36-cp36m-manylinux1_x86_64.whl (361kB)
    100% |████████████████████████████████| 368kB 6.1MB/s
Requirement already satisfied: numpy>=1.6.1 in /usr/local/lib/python3.6/dist-packages (from
WordCloud) (1.14.6)
Requirement already satisfied: pillow in /usr/local/lib/python3.6/dist-packages (from WordCloud)
(4.0.0)
Requirement already satisfied: olefile in /usr/local/lib/python3.6/dist-packages (from pillow-
>WordCloud) (0.46)
Installing collected packages: WordCloud
Successfully installed WordCloud-1.5.0
```

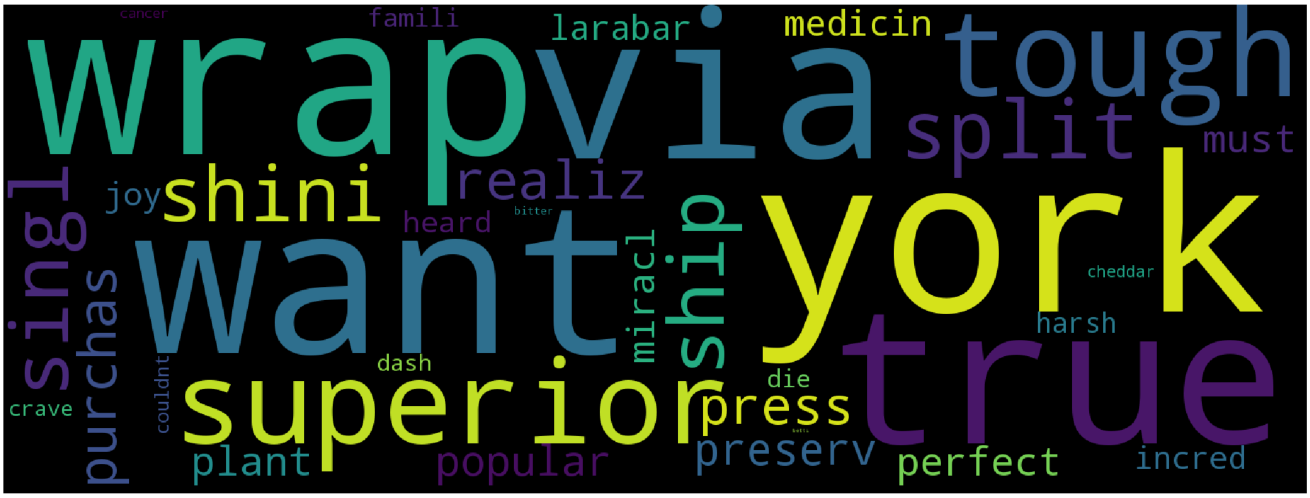
In [0]:

```
from wordcloud import WordCloud
value=[]
for i in range(len(cluster30)):
  value.append(i)


result = dict(zip(cluster30,value))
#dict(result.items())
```

In [0]:

```
from wordcloud import WordCloud
import matplotlib.pyplot as plt
```

```
# Lets first convert the 'result' dictionary to 'list of tuples'
tup = dict(result.items())
#Initializing WordCloud using frequencies of tags.
wordcloud = WordCloud(     background_color='black',
                           width=1600,
                           height=600,
                     ).generate_from_frequencies(tup)

fig = plt.figure(figsize=(30,20))
plt.imshow(wordcloud)
plt.axis('off')
plt.tight_layout(pad=0)
fig.savefig("tag.png")
plt.show()
```



Observation:

- From the word cloud of cluster30 we can observe it mostly relates to taste

Cosine smilarity of the vectors

In [0]:

```
# Remembering TSVD formulea:

#             TSVD      =    U          * Sigma           * V.T
#                 nXn        nxk                kxk              kxn

#
```

In [0]:

```
tsvd1= TruncatedSVD(n_components=500)
data=tsvd1.fit_transform(mat)

dictionary = dict(zip(m.get_feature_names(), tsvd1.components_))

df=pd.DataFrame.from_dict(dictionary,orient='index')
```

In [0]:

```
names= df.index.values
```

In [0]:

```
from scipy.spatial.distance import pdist, squareform
```

In [0]:

```python
from sklearn.metrics.pairwise import cosine_similarity

#cosine_similarity(df.sum(axis=0).values)


dict(zip(df.index,cosine_similarity(df)))

names1=[]
for a in names:
  names1.append(a)
```

In [0]:

```python
final=dict(zip(df.index,cosine_similarity(df)))
df1=pd.DataFrame.from_dict(final,orient='index')
df2=pd.DataFrame(data=final,columns=[names1])
```

In [0]:

```python
mat1 = pd.DataFrame(final)
mat1.index = names1
mat1
```

Out[0]:

	abl	absolut	absorb	acai	accept	accord	acid	a
abl	1.000000e+00	-1.322727e-17	1.893559e-16	-1.405126e-16	1.110223e-16	-1.040834e-17	1.734723e-16	5.247539
absolut	-1.322727e-17	1.000000e+00	-2.350550e-16	-3.382711e-17	2.798326e-16	2.905662e-17	-7.979728e-17	1.973248
absorb	1.893559e-16	-2.350550e-16	1.000000e+00	1.591609e-16	-3.729655e-17	-4.722785e-16	-3.677614e-16	9.601694
acai	-1.405126e-16	-3.382711e-17	1.591609e-16	1.000000e+00	-2.409097e-16	1.188286e-16	-1.890849e-16	-2.32452 16
accept	1.110223e-16	2.798326e-16	-3.729655e-17	-2.409097e-16	1.000000e+00	2.706169e-16	4.965646e-16	-6.87817 16
accord	-1.040834e-17	2.905662e-17	-4.722785e-16	1.188286e-16	2.706169e-16	1.000000e+00	5.637851e-17	4.554733
acid	1.734723e-16	-7.979728e-17	-3.677614e-16	-1.890849e-16	4.965646e-16	5.637851e-17	1.000000e+00	1.351350
acquir	5.247539e-16	1.973248e-16	9.601694e-16	-2.324529e-16	-6.878179e-16	4.554733e-16	1.351350e-15	1.000000
across	7.285839e-17	5.724587e-17	6.578939e-16	6.080206e-16	1.910568e-16	-1.205633e-16	9.185903e-16	-6.19296 16
act	-2.758210e-16	-1.344411e-16	-7.546047e-17	4.163336e-17	1.383442e-16	3.328501e-17	-3.816392e-17	9.922618
activ	1.476683e-16	4.683753e-17	3.794708e-16	5.186823e-16	9.801188e-17	-1.214306e-16	5.919744e-16	-8.25511 16
actual	1.986258e-16	2.827599e-16	1.543904e-16	-1.925543e-16	3.122502e-16	-3.382711e-17	-4.007211e-16	4.302114
ad	3.885781e-16	9.020562e-17	1.647987e-16	1.665335e-16	5.542442e-16	-2.322903e-17	3.608225e-16	1.535230
add	1.138412e-16	-3.001072e-16	-2.519686e-16	-1.054929e-16	-3.113829e-16	6.383782e-16	2.029626e-16	-3.87277 16
addict	-4.749890e-16	1.647987e-16	9.194034e-17	1.613293e-16	9.194034e-17	9.378349e-17	2.255141e-17	1.075529
addit	-1.066855e-16	-3.929149e-16	1.222980e-16	-2.914335e-16	-1.717376e-16	-4.147073e-16	-2.029626e-16	4.302114
	-1.634977e-	-1.700029e-	-3.295975e-	-5.113097e-				-8.50014

	abl	absolut	absorb	acai	accept	accord	acid	a
adjust	16	16	16	16	1.274785e-17	1.361758e-16	9.714451e-17	17
admit	-1.734723e-16	-1.405126e-16	-2.576064e-16	1.890849e-16	2.272488e-16	6.852158e-17	1.509209e-16	-6.24500 17
ador	-5.377643e-17	1.214306e-16	8.283305e-17	-3.521489e-16	-3.469447e-17	1.379105e-16	3.452100e-16	-1.52655 16
adult	-3.382711e-17	1.592151e-16	5.009014e-17	1.144917e-16	-4.857226e-17	3.217912e-16	1.335737e-16	1.70870
advantag	-9.020562e-17	-2.081668e-17	-1.569925e-16	-6.370772e-16	3.426079e-16	-3.226586e-16	-1.517883e-16	1.55908
advertis	3.070461e-16	-1.084202e-16	-4.805184e-16	7.112366e-17	-1.856154e-16	1.101549e-16	-2.597748e-16	4.25007
advic	-3.859760e-16	-6.765422e-17	0.000000e+00	-3.144186e-17	1.567756e-16	3.365364e-16	3.035766e-17	-5.82867 16
advis	4.119968e-18	-4.510281e-17	1.257675e-16	2.411266e-16	4.562323e-16	-9.801188e-17	-4.250073e-16	3.33934
affect	4.475587e-16	-4.336809e-17	-1.979753e-16	8.890458e-17	-2.498002e-16	2.020953e-16	-4.544976e-16	1.23165
afford	7.979728e-17	2.220446e-16	-3.816392e-17	8.196568e-17	-1.450663e-16	3.295975e-17	-4.857226e-17	-1.95156 18
afraid	-8.326673e-17	-1.222980e-16	1.279359e-16	-3.226586e-16	2.324529e-16	2.299593e-16	-2.151057e-16	1.45716
afternoon	-1.040834e-16	-1.353084e-16	-8.673617e-18	-2.528359e-16	-3.885781e-16	-2.779894e-16	1.032160e-16	8.15320
aftertast	2.671474e-16	5.941428e-17	-3.469447e-18	-6.938894e-17	-1.778092e-16	1.222980e-16	-5.811324e-17	2.65412
afterward	-7.936360e-17	7.654467e-17	-4.336809e-17	1.370432e-16	-3.174544e-16	-3.295975e-17	-2.168404e-18	-6.59194 17
...	...	...	...	...	...	...	...	...
desert	-1.314053e-16	-1.110223e-16	-9.367507e-17	-1.658829e-17	6.852158e-17	-4.597017e-17	-8.782038e-17	-9.97466 17
design	-4.163336e-17	4.618701e-17	-1.474515e-17	-1.040834e-17	2.775558e-17	-1.908196e-17	-3.773024e-17	-7.45931 17
desir	7.546047e-17	5.204170e-17	4.857226e-17	-9.107298e-17	-3.035766e-17	3.382711e-17	5.377643e-17	3.38271
desk	-2.255141e-17	2.084379e-17	1.387779e-17	6.938894e-17	-3.599551e-17	-2.255141e-17	-6.028164e-17	1.73472
despit	-5.030698e-17	-3.079134e-17	1.908196e-17	2.168404e-17	-1.734723e-18	-6.765422e-17	-8.673617e-18	3.98986
dessert	-8.630249e-17	-1.734723e-17	-5.811324e-17	-4.163336e-17	1.214306e-17	-6.114900e-17	2.385245e-18	4.59701
destroy	-1.665335e-16	-1.604619e-17	7.285839e-17	5.204170e-18	-7.806256e-18	-1.249001e-16	-3.588709e-17	-1.22514 17
detail	-4.342230e-17	2.602085e-17	-3.295975e-17	-2.081668e-17	-2.233456e-17	2.078619e-17	8.066464e-17	-3.72965 17
develop	6.938894e-17	5.551115e-17	1.366095e-17	-3.469447e-17	-2.840610e-17	2.905662e-17	4.987330e-17	1.73472
diabet	-4.987330e-18	-2.775558e-17	-4.510281e-17	-2.428613e-17	6.245005e-17	-1.019150e-17	-3.816392e-17	2.16840
diagnos	7.892992e-17	2.775558e-17	-9.410875e-17	4.597017e-17	-8.673617e-19	1.682682e-16	-1.092876e-16	-1.43114 16
diarrhea	-7.806256e-18	-3.469447e-18	1.734723e-17	3.035766e-17	1.691355e-17	-1.977314e-17	-6.174531e-17	3.46944
[illegible]	1.778092e-17	2.081668e-17	-8.673617e-	2.341877e-17	-4.293441e-	8.673617e-19	1.821460e-17	-2.94903

	abl	absolut	absorb	acai	accept	accord	acid	a
didnt	1.734723e-17	6.938894e-18	3.469447e-18	-2.949030e-17	1.301043e-17	4.336809e-18	1.658152e-17	2.341877
die	-5.204170e-18	-1.452831e-16	5.551115e-17	-4.813858e-17	5.551115e-17	-1.489694e-16	3.642919e-17	4.163336
diet	5.204170e-18	-1.647987e-17	9.540979e-18	-1.387779e-17	7.285839e-17	-1.318390e-16	2.775558e-17	5.518589
dietari	4.033232e-17	-8.153200e-17	1.647987e-17	1.322727e-17	1.886512e-17	-7.112366e-17	-3.903128e-17	-5.20417 18
differ	5.984796e-17	-8.500145e-17	1.078781e-17	-1.474515e-17	7.025630e-17	-6.071532e-17	-6.776264e-18	-5.03069 17
difficult	-5.377643e-17	5.854692e-17	-5.681219e-17	4.770490e-18	6.461845e-17	-2.602085e-17	1.214306e-17	-3.81639 17
digest	4.076600e-17	4.683753e-17	1.908196e-17	3.295975e-17	4.987330e-17	-1.301043e-17	-5.290907e-17	2.471981
dilut	1.301043e-18	-3.758455e-17	1.006140e-16	-1.127570e-17	-6.245005e-17	-8.673617e-19	8.890458e-18	-7.80625 18
dinner	-1.214306e-17	2.862294e-17	3.295975e-17	-2.428613e-17	7.806256e-18	-3.007306e-17	-4.770490e-17	-1.04083 17
dip	-2.558717e-17	4.336809e-17	-3.469447e-18	2.428613e-17	4.510281e-17	4.363914e-17	1.908196e-17	-6.74373 17
direct	-3.469447e-18	3.209238e-17	9.844556e-17	3.697807e-17	2.428613e-17	-2.585822e-17	1.994932e-17	-1.91903 17
dirti	1.130281e-17	3.198396e-17	1.778092e-17	-2.775558e-17	-2.428613e-17	-3.035766e-17	3.469447e-18	2.428613
disappear	-8.500145e-17	-1.043002e-16	-6.765422e-17	-1.110223e-16	1.734723e-17	-1.040834e-16	1.682682e-16	3.122502
disappoint	1.387779e-17	-1.474515e-17	-1.214306e-17	-1.214306e-17	-3.382711e-17	-1.322727e-17	-5.746272e-18	-4.33680 17
discontinu	-1.539567e-17	-5.247539e-17	-4.943962e-17	-6.331741e-17	1.040834e-17	-4.293441e-17	2.255141e-17	4.076600
discount	8.066464e-17	-4.401861e-17	3.122502e-17	1.131907e-16	-3.295975e-17	5.204170e-18	-5.139118e-17	-3.90312 18
discov	-3.122502e-17	1.301043e-16	1.864828e-17	-1.244664e-16	5.724587e-17	4.943962e-17	-2.298509e-17	3.816392

500 rows × 500 columns

In [0]:

```
from sklearn.metrics.pairwise import cosine_similarity
from scipy import sparse
```

In [0]:

```
######################################################
# Just for my understanding
= cosine_similarity(df)
A_sparse = sparse.csr_matrix(A)

similarities_sparse = cosine_similarity(A_sparse,dense_output=False)
print('pairwise sparse output:\n {}\n'.format(similarities_sparse))
########################################################
```

In [0]:

```
#Cosine similarity from randomised svd
from sklearn.utils.extmath import randomized_svd
```

```
# Matrix factorization X = U Σ V.T
U1, Sigma1, VT1 = randomized_svd(mat.values,
                              n_components=500,
                              n_iter=10,
                              random_state=None)
```

In [37]:

```
# Standardizing data with mean=0 and variance = 1 on U1
from sklearn.preprocessing import StandardScaler
standardized_data_tf_idf_kd = StandardScaler(with_mean=False).fit_transform(U1)
print(standardized_data_tf_idf_kd.shape)
```

```
(2000, 500)
```

In [0]:

```
final= np.array(standardized_data_tf_idf_kd) # storing the values after standardization in a dense
array
```

In [0]:

```
# Processing words in Kmeans++
```

In [39]:

```
from sklearn.cluster import KMeans

# applying Kmeans++ on 25 clusters
kmeans_1 = KMeans(n_clusters=25, random_state=42, n_init=30, n_jobs=-1)
kmeans_1.fit(final)
```

Out[39]:

```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
    n_clusters=25, n_init=30, n_jobs=-1, precompute_distances='auto',
    random_state=42, tol=0.0001, verbose=0)
```

In [65]:

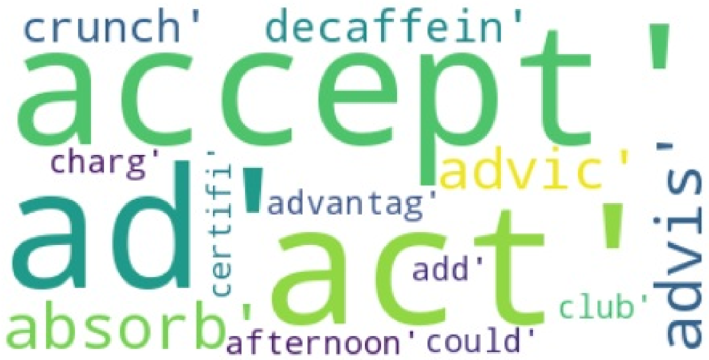
```
from wordcloud import WordCloud,STOPWORDS
import matplotlib.pyplot as plt
stopwords_t=set(STOPWORDS)

kmeans_1.predict(final) # for labels
centroids=kmeans_1.cluster_centers_.argsort() # centers of clusters
terms = m.get_feature_names() # IDF_ words


list3 = []

for i in range(15):
    print("Cluster %d:" % i, end='')
    for j in centroids[i, :15]:
        list3.append(terms[j])
    wc = WordCloud(background_color="white", max_words=len(str(list3)), stopwords=stopwords_t)
    wc.generate(str(list3))
    print("Cosine Similarity of cluster:", i)
    plt.imshow(wc, interpolation='bilinear')
    plt.axis("off")
    plt.show()
    list3.clear()
```

```
Cluster 0:Cosine Similarity of cluster: 0
```

Cluster 1:Cosine Similarity of cluster: 1

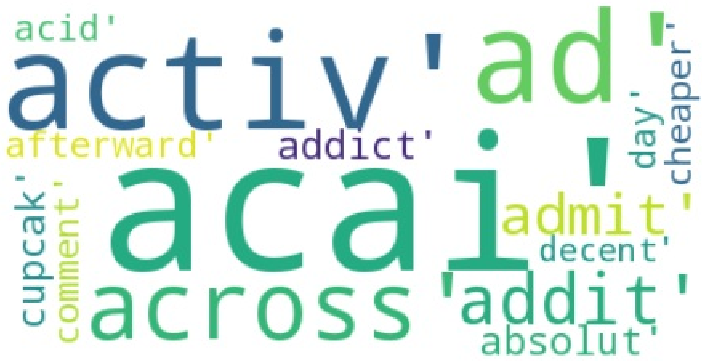
Cluster 2:Cosine Similarity of cluster: 2

Cluster 3:Cosine Similarity of cluster: 3

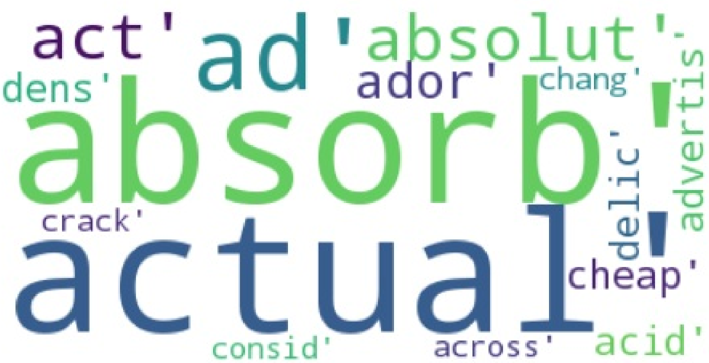


Cluster 4:Cosine Similarity of cluster: 4

Cluster 5:Cosine Similarity of cluster: 5

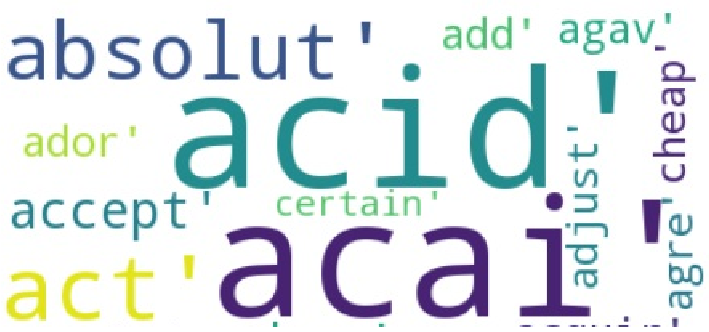


Cluster 6:Cosine Similarity of cluster: 6

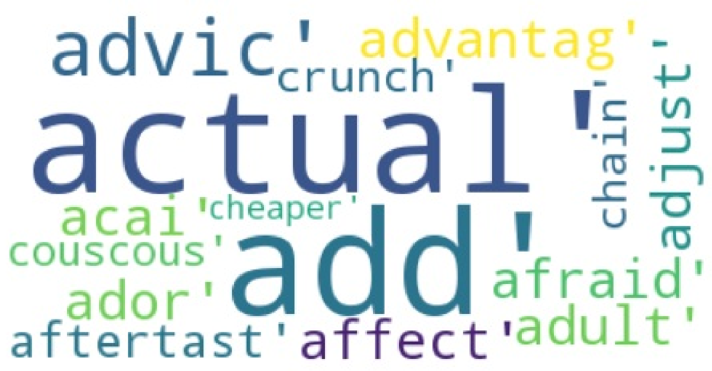


Cluster 7:Cosine Similarity of cluster: 7

Cluster 8:Cosine Similarity of cluster: 8

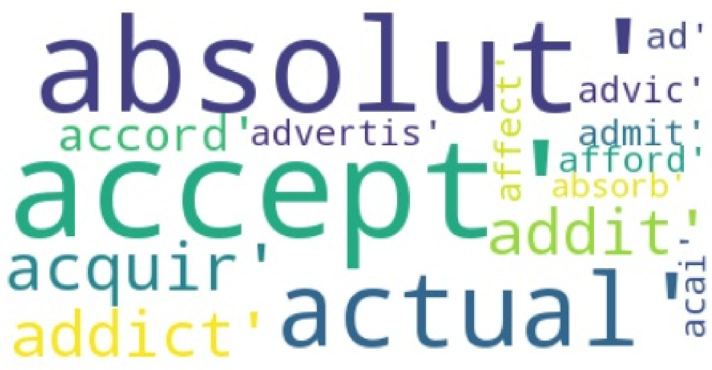


Cluster 9:Cosine Similarity of cluster: 9

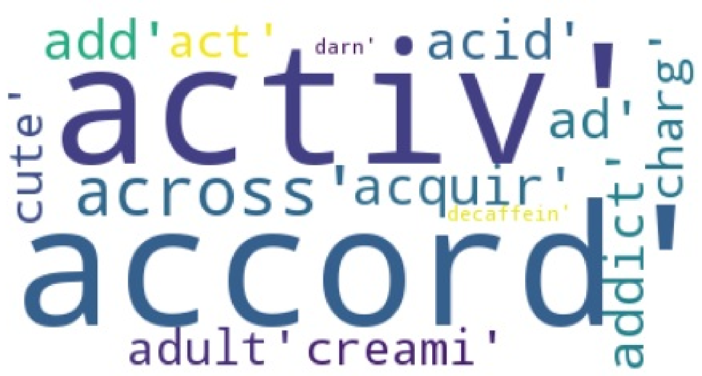


Cluster 10:Cosine Similarity of cluster: 10

Cluster 11:Cosine Similarity of cluster: 11

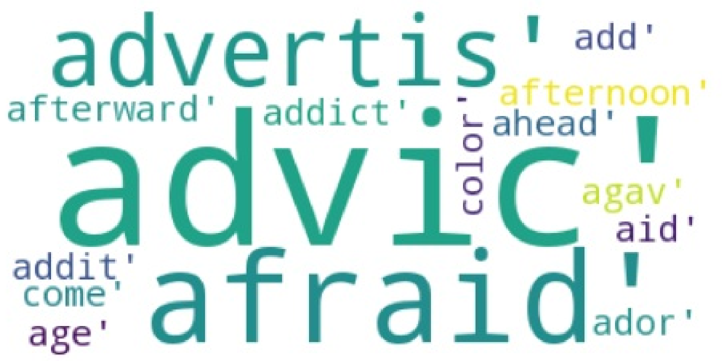


Cluster 12:Cosine Similarity of cluster: 12



```
Cluster 13:Cosine Similarity of cluster: 13
```


```
Cluster 14:Cosine Similarity of cluster: 14
```



Observation:

In the top clusters found below words have more cosine similarity

1. acai' ~ absorb
2. act~active
3. admit~ addit
4. accept~ad~act